高性能服务器编程目录

一 基本方法篇

0 引言

high-performance server architecture--Jeff Darcy

总结分析了影响服务器性能的四大杀手：

- ✓ 数据拷贝（Data Copies）；
- ✓ 环境切换（Context Switches）；
- ✓ 内存分配（Memory allocation）；
- ✓ 锁竞争（Lock contention）。

1 网络模型

- ✓ 阻塞/非阻塞；
- ✓ 同步/异步；
- ✓ IO 复用；
- ✓ 长连接/短连接

1.1 The C10K problem

总结了并发上万连接的常用高性能 IO 策略：

- ✓ 一个线程服务多个客户端，使用非阻塞 I/O 和水平触发的就绪通知；
- ✓ 一个线程服务多个客户端，使用非阻塞 I/O 和就绪改变时通知；
- ✓ 一个服务线程服务多个客户端，使用异步 I/O；
- ✓ 一个服务线程服务一个客户端，使用阻塞 I/O；
- ✓ 把服务代码编译进内核。

知识点：

- ✓ **select/ poll/ kqueue**；
- ✓ **epoll**：水平出发 vs 边缘触发
- ✓ **完成端口 IOCP**
- ✓ NPTL；
- ✓ Zero-copy；
- ✓ sendfile；
- ✓ writev/readv。

1.2 异步 IO

- ✓ Linux epoll--异步阻塞
- ✓ Linux aio--异步非阻塞
- ✓ Windows iocp--异步非阻塞
- ✓ Boost asio

1.3 Topics in High-Performance Messaging

讨论了 TCP 延时和 UDP 缓冲，以及流量控制的一些话题：

- ✓ TCP Latency

- ✓ **UDP Buffer**
- ✓ Multicast

2 多线程多进程

2.1 常用服务器模型：

- ✓ 单进程（线程）迭代
- ✓ 父进程+动态 fork 子进程（线程）
- ✓ 预创建进程（线程）池
- ✓ 父进程+预创建进程（线程）池
- ✓ 单进程 IO 复用
- ✓ 单进程接入+处理子进程（线程）池
- ✓ 单进程接入+少数处理进程（线程）

2.2 NPTL

阐述了 linux 多线程库 NPTL 的设计：

- ✓ 1 on 1 vs M on N
- ✓ No manager thread

2.3 多线程模型

- ✓ 线程特定存储-Google 的内存分配器 TCMalloc

2.4 CPU 亲和力

多核体系问题：线程间切换开销；多核体系处理器 cache 的"乒乓效应"。CPU 亲和力指在多核体系中进程（线程）能够单独绑定到一个处理核的能力。使用硬亲合力的场景有：

- ✓ 有大量计算要做
- ✓ 复杂的应用程序
- ✓ 时间敏感的、决定性的进程

3 锁相关技术

- ✓ 进程锁 vs 线程锁
- ✓ 加锁的粒度：库级锁/表级锁/页级锁/行级锁
- ✓ 各种锁系统开销：互斥锁/读写锁/意向锁/自旋锁
- ✓ 分布式加锁算法
- ✓ 锁表的实现
- ✓ Lock free
- ✓ 免锁模型：环形缓冲区

4 内存磁盘存储模型

4.1 减少数据拷贝

4.2 池式技术

- ✓ Memory pool
- ✓ Object pool

4.3 读写技术

- ✓ **Prefetch**
- ✓ **Delay write**
- ✓ **Copy On Write**
- ✓ **空间换时间**
- ✓ **双缓冲技术**

5 通信协议

5.1 Google protocol buffer

- 序列化和反序列化
- 一种描述数据格式的 IDL，生成了更容易在编程中使用的数据访问类
- 对 IDL 所描述的格式进行编码的一种二进制编码方案
- 通过代码生成器实现的数据绑定支持，Google 提供了 C++、Python、Java 实现
- 与 XML 相比,Protocol Buffers 文件的尺寸要小3-10倍,解析速度要快20-100倍.

5.3 ICE/Corba 等 IDL

6 有状态服务/无状态服务器

有状态服务器：

- ✓ 服务器中保存少量信息可以减少客户和服务器之间交换报文的大小，快速响应请求。
- ✓ 应用协议规定某个报文取决于先前的报文
- ✓ 维护**状态机**，实现复杂

无状态服务器：

- ✓ 报文既不能依赖是否被按序交付，也不能依赖先前已经被交付的报文
- ✓ 实现简单高效

二 典型网络模式

1 Reactor-同步

2 Proactor-异步

3 半同步半异步

4 领导者/追随者

5 A Design Framework for Highly Concurrent Systems

在 tasks, queues, thread pools 等设计元素的基础上提出了常用的高并发系统设计模式。

三 系统实现篇

1 libevent

libevent 是一个事件触发的网络库，适用于 windows、linux、bsd 等多种平台，内部使用 select、epoll、kqueue 等系统调用管理事件机制。典型 Reactor 模式。

2 spserver

spserver 是一个实现了半同步/半异步(Half-Sync/Half-Async)和领导者/追随者(Leader/Follower) 模式的服务器框架，网络模块采用 libevent。

3 memcached 网络模块

高效率的 key-value 内存存储系统，网络模块采用 libevent 实现。

4 ACE

重量级的网络 IO 框架，学院派的 ACE 总结的网络模型常用设计模式值得研究和学习。

5 Flash: An Efficient and Portable Web Server